

Live model transformations driven by incremental pattern matching

István Ráth (rath@mit.bme.hu)

Gábor Bergmann

András Ökrös

Dániel Varró

Overview

- Introduction
- Live transformations
- The VIATRA implementation
- Future work
- Summary

INTRODUCTION

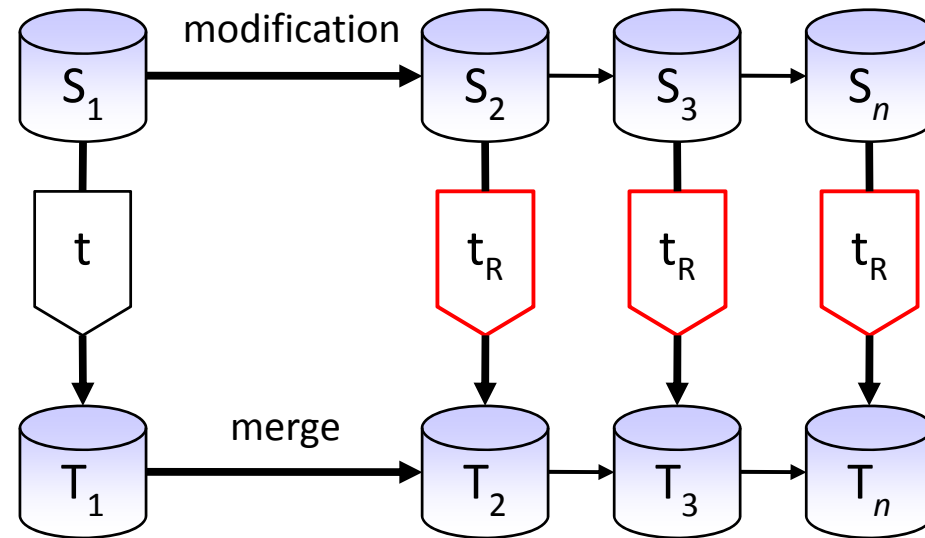
Model transformations and incrementality

- Key usage scenarios for MT:
 - Mapping between languages or various levels of abstraction
 - Intra-domain model manipulation
- Working with **evolving** models
 - Constant changes & modifications
 - **Large** models
- Problem: (re)transformations are slow
 - To execute... (large models)
 - and to re-execute again and again (batch execution).
- Solution: **incrementality**
 - Take the source model, and its mapped counterpart;
 - Use the information about how the source model was changed;
 - Map and apply the changes (but **ONLY** the changes) to the target model.

Towards incrementality

- How to achieve incrementality?
 - Incremental *updates*: avoid re-generation.
 - Don't recreate what is already there.
 - → Use reference (correspondence) models.

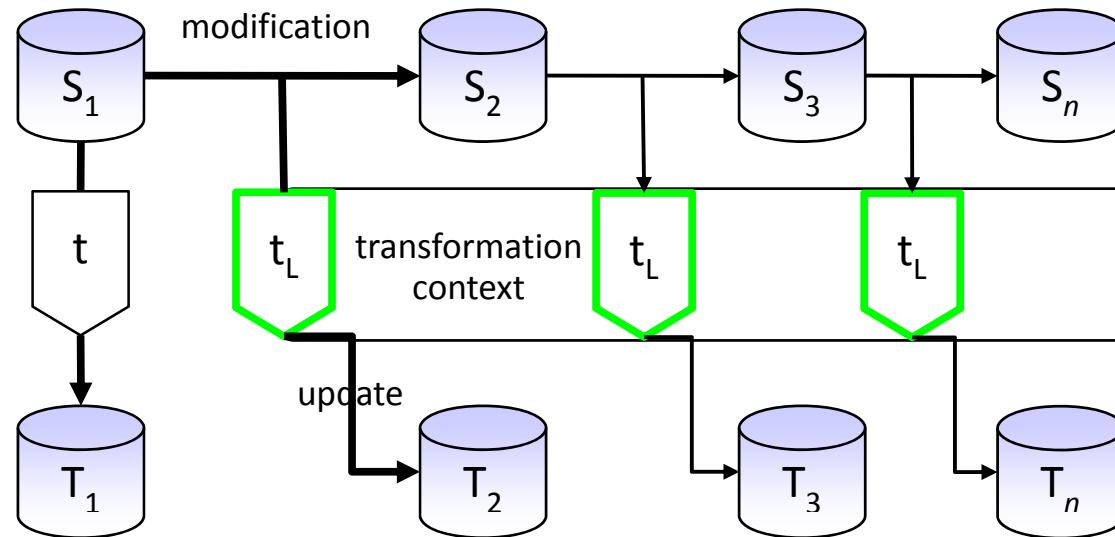
Incremental updates



Towards incrementality

- How to achieve incrementality?
 - Incremental *updates*: avoid re-generation.
 - Don't recreate what is already there.
 - → Use reference (correspondence) models.
 - Incremental *execution*: avoid re-computation.
 - Don't recalculate what was already computed.
 - How?

Live transformations⁰



⁰ Hearnden-Lawley-Raymond. *Incremental Model Transformation for the Evolution of Model-driven Systems*. MODELS 2006.

Live transformations?

- Essentially: a different execution mode.
 - More than “simple” incremental execution...
- Run as “daemons”
 - whenever model changes occur
- Maintain a *transformation context*
 - To persist (cache) the execution state
- Model changes can be instantly mapped
 - Input
 - *what* has changed (creation, deletion)
 - *where* has the change occurred (source, reference, target)
 - Output:
 - *actions* (what to do)

Possible applications

- M2M / inter-domain transformations
 - Model synchronization on-the-fly
- M2M / **intra**-domain transformations
 - Model animation/execution (based on dynamic semantics)
 - Constraint checking on-the-fly (what-if)
- M2C
 - Incremental model-code synchronization (DSMLs)
- ...

GRAPH TRIGGERS IN VIATRA2

Graph transformations

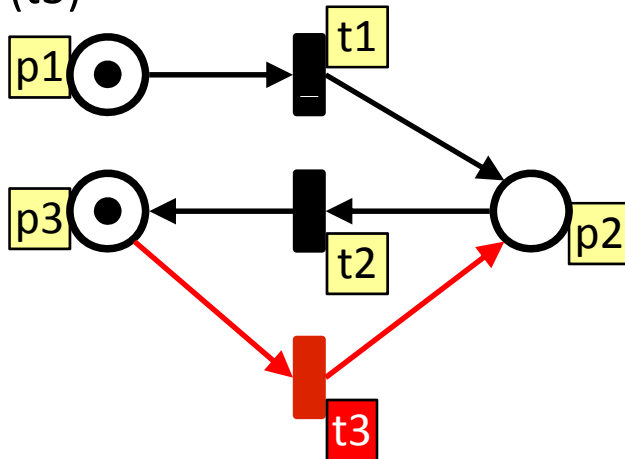
- Graph rewriting based on rules
- Driven by *graph patterns*
 - Subgraph to be matched in the model graph
 - A set of constraints (type, attribute value, structural, ...)
- Execution
 - Declarative approaches (GT rules, TGG)
 - Imperative (mixed) approaches (explicit control flow)

Incremental graph pattern matching

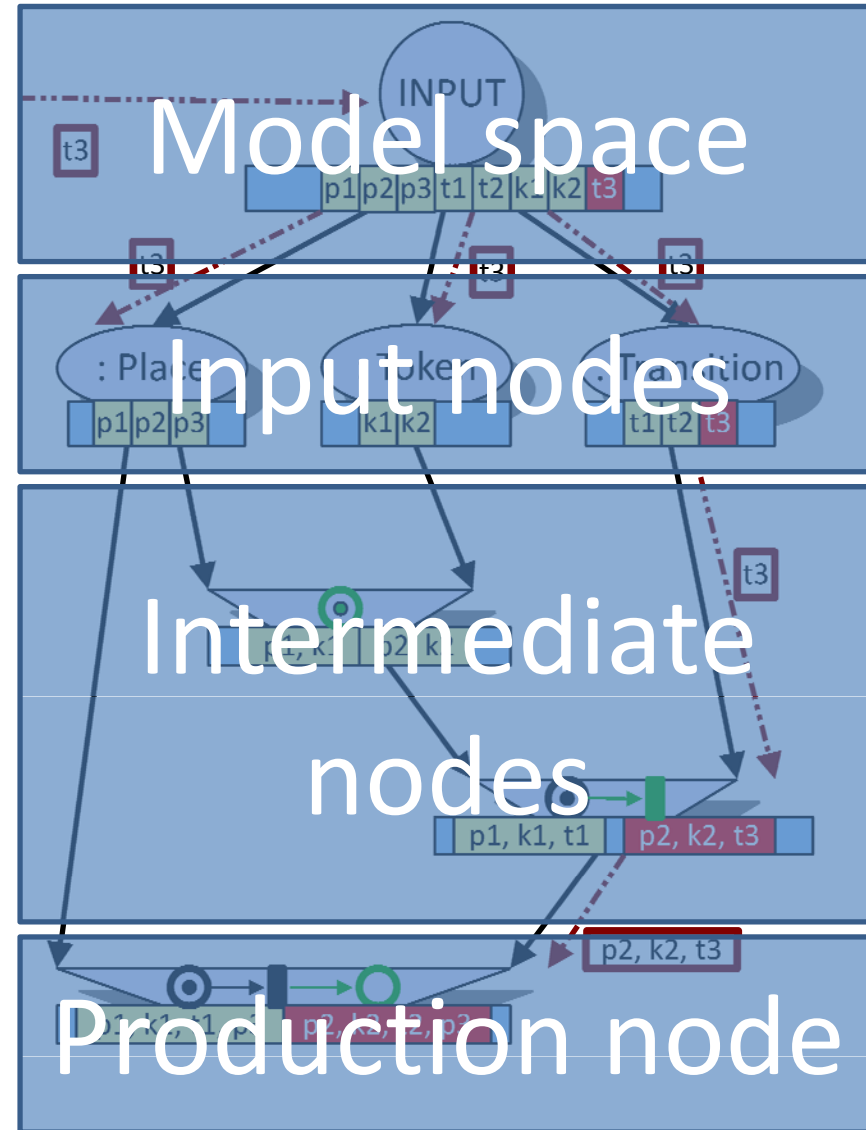
- Graph transformations require pattern matching
- Goal: retrieve the *matching set* quickly
- How?
 - Store (cache) matchings → execution context
 - Update them as the model changes
 - Update precisely → **incrementality**

Core idea: use RETE¹ nets

- RETE network
 - node: (partial) matches of a (sub)pattern
 - edge: update propagation
- Demonstrating the principle
 - input: Petri net
 - pattern: fireable transition
 - Model change: new transition (t3)



¹ Forgy, C.L.: RETE: A fast algorithm for the many pattern/many object pattern match problem. AI 1982.



Updates

- Needed when the model space changes
- VIATRA notification mechanism (EMF is also possible)
 - Transparent: user modification, model imports, results of a transformation, external modification, ... → RETE is always updated!
- Input nodes receive elementary modifications and release an update token
 - Represents a change in the partial matching (+/-)
- Nodes process updates and propagate them if needed
 - PRECISE update mechanism

Event-driven live transformations

- An idea: represent events as model elements.²
- Our take: represent events as changes in the matching set of a pattern.
 - ~generalization
- Incrementality: persistent context
 - matching sets cached in RETE networks → **efficiency!**
 - global cache (“session”)
- Formalism
 - Specify WHEN and HOW to (re)act
 - Should be “MT-like”

² E. Guerra – J. de Lara. *Event-Driven Grammars: Towards the Integration of Meta-modelling and Graph Transformation*, 2004

Formalism

```
@trigger(sensitivity="rise", priority="1", mode="always")
```

```
gtrule mapClass() =
```

```
{
```

High level support for:

- Model element **creation/deletion**
- **Attribute value changes** (special language constructs)
- **Arbitrarily complex changes** (involving multiple model elements)
- Model changes from **any source** (user editing, transformations, model imports, ...)

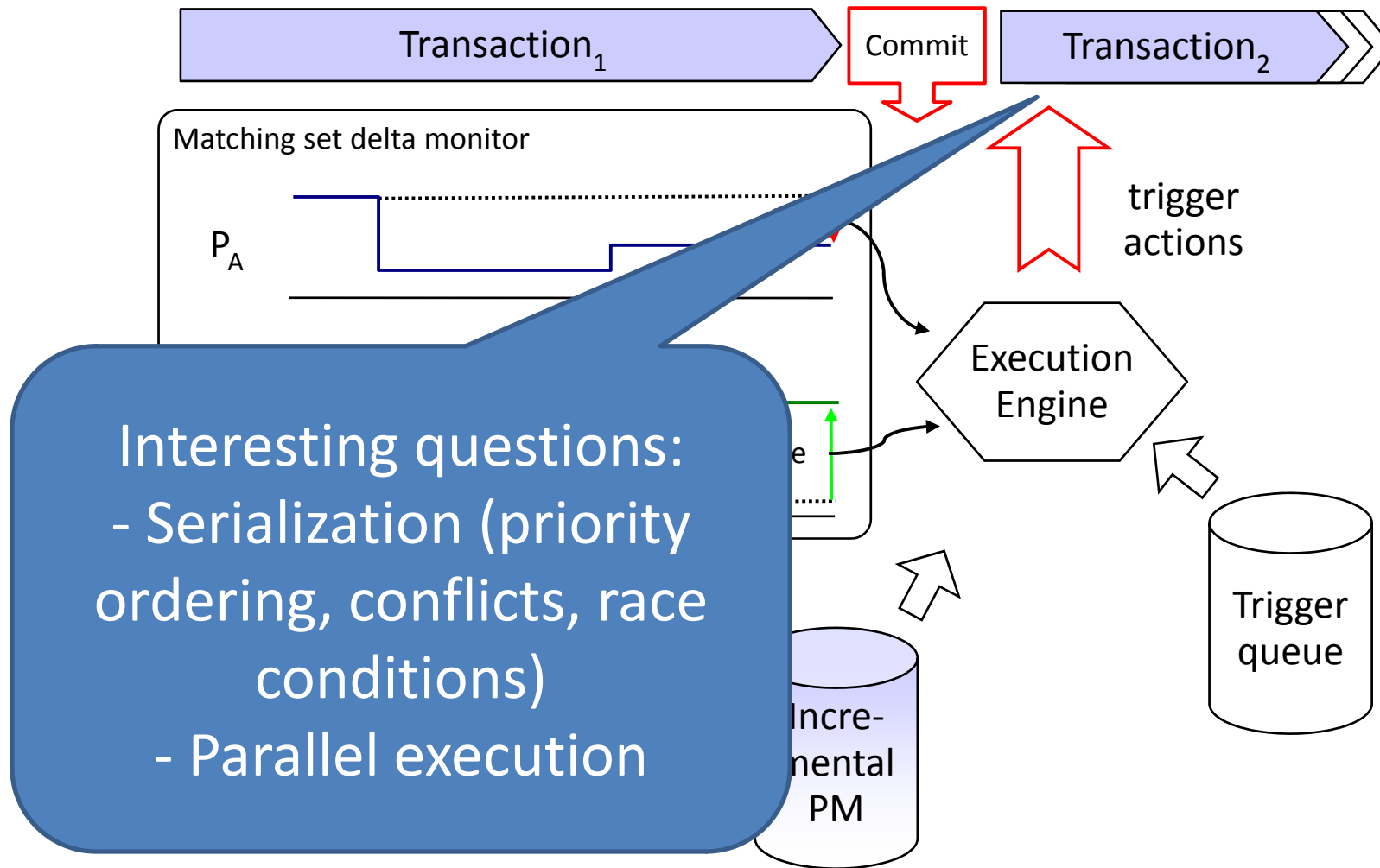
```
}
```

Annotations to specify execution modes

the graph pattern whose binding set changes trigger

sequence: imperative to
model modification, debug
printing, code generation, ...

Execution



APPLICATIONS & FUTURE WORK

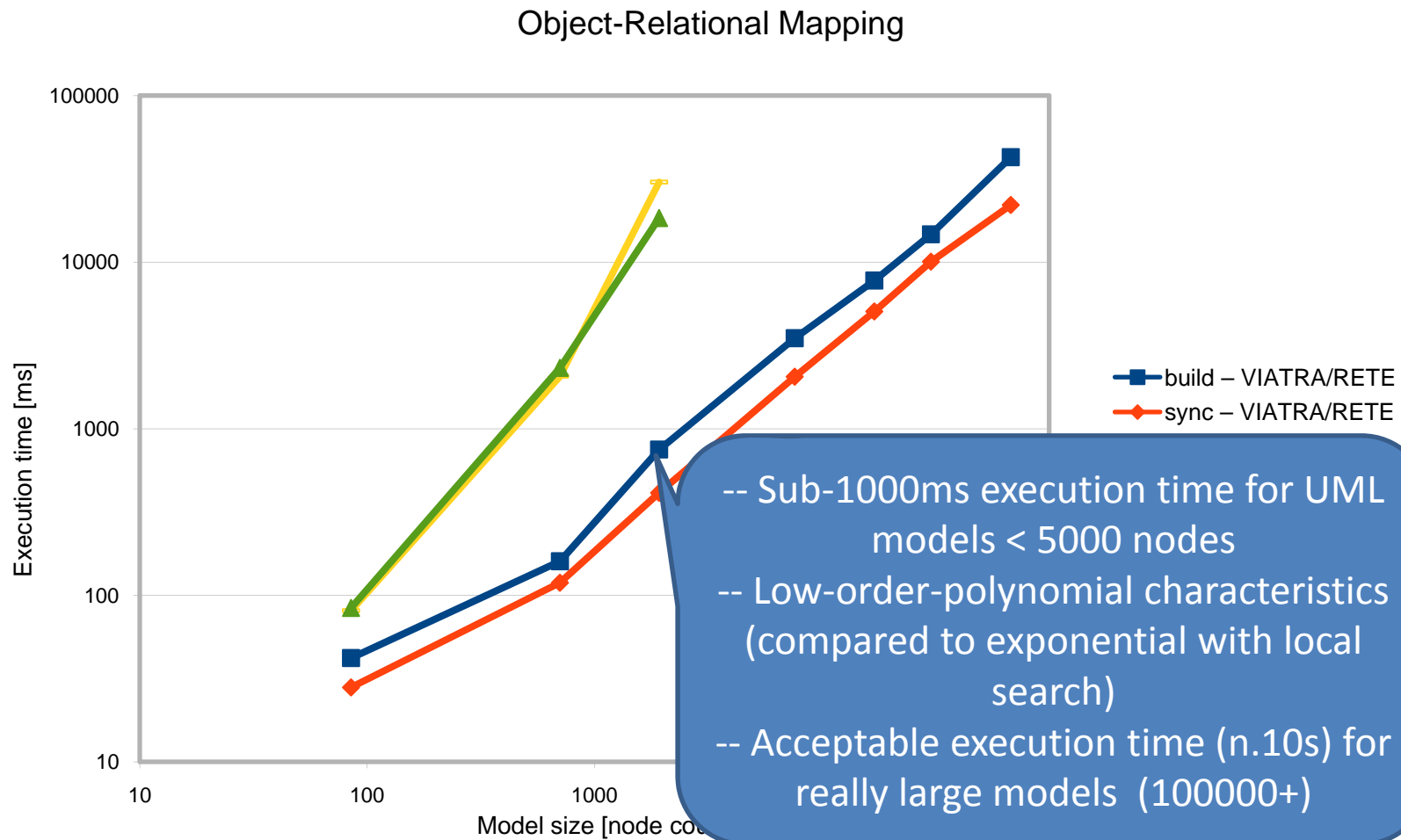
Case studies

- Domain-specific visual languages (ViatraDSM)
 - Incremental, on-the-fly constraint evaluation (& enforcement)
 - On-the-fly abstraction
 - Metamodel-driven
 - Incremental code generation
 - Model-code synchronization
 - Model animation & simulation

-- VIATRA2 is not limited to the „source→target” scenario
-- can be used to map a DSML into multiple concrete syntax representations (analysis models, architectural models, etc)
simultaneously, on multiple levels of abstraction

³ Ráth-Vágó-Varró. *Design-time simulation of domain-specific models by incremental pattern matching*. Accepted for VL/HCC 2008.

Performance⁴



⁴ Bergmann-Horváth-Ráth-Varró. *A benchmark evaluation of incremental pattern matching in graph transformation*. To appear in Proc. ICGT2008.

Future work

- Improving performance
 - Parallel (multi-core) execution
 - RETE optimizations
- Language improvements
 - More compact notations
 - Static type checking
 - Debugging support
- EMF support
 - Incremental pattern matching & matching set detection (EMFT)
 - Java API based on the VIATRA language

Conclusion

- Live transformation engine for incremental graph transformations
 - Novel approach based on matching set changes, supported by the new, high performance incremental pattern matching engine
- VIATRA2 R3
 - <http://eclipse.org/gmt>
 - <http://wiki.eclipse.org/VIATRA2>
 - Available as a preview release (coming in August)
- Thank you.